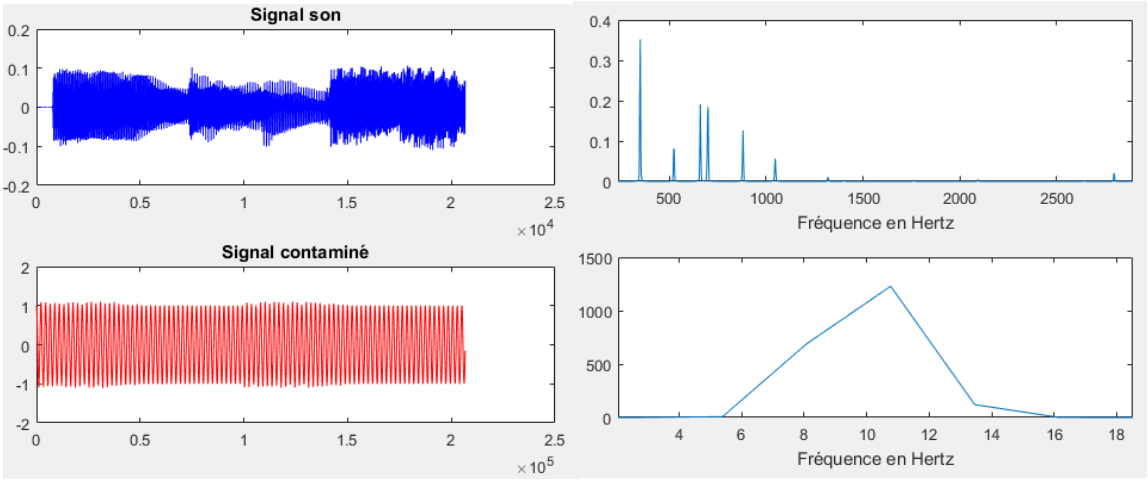
Mattéo Paternotte

TP

Traitement numérique du son

20.

```
clear all;
nfft= 8192
f= 10 %fréquence de la siusoide
[x,Fe]= audioread('Ring03.wav'); %lecture du fichier son
sound(x,Fe) %ecoute du fichier son
Te= 1/Fe
N= length (x)-1;
T= N*Te %duré du son
t= 0:Te:T; %tableau des temps
s= cos(2*pi*f*t); %signal sinusoidal
y= x'+s; %contamination
sound(y,Fe);%ecoute du signal contaminé
%caractéristiques dynamiques
Dx= abs(max(x)-min(x))
dymx= 20*log10(Dx)
Dy= abs(max(y)-min(y))
dymy= 20*log10(Dy)
%caracteristique energetique
X= max(x).*max(x)
Y= max(y).*max(y)
energieX= sum(x.*x)/X%valeur normalisées
energieY= sum(y.*y)/Y
%allure temporelle
figure(1);
subplot(2,1,1);plot(x(1:N/10),'b');title('Signal son')
subplot(2,1,2); plot(y(1:N/1),'r');title('Signal contaminé')
%analyse spectrale
[pxx,f]= psd(x,nfft,Fe);
[pxy,f]= psd(y,nfft,Fe);
%représentation des spectre d'amplitude des signaux
figure(2);
subplot(2,1,1);plot(f,pxx);xlabel('Fréquence en Hertz')
subplot(2,1,2); plot(f,pxy);xlabel('Fréquence en Hertz')
%fréquence de l'amplitude max du spectre
[Y,I]= max(pxx)
fremaxx=f(I)%frequence max damplitude du son
[Y,I]= max(pxy)
fremaxy=f(I)%frequence max damplitude du son contaminé
```



21.

a)

```
%effet echo simple
[x,fs]= audioread('ring03.wav');%lecture fichier.wav
sound(x,fs); %écoute du son original
N= length(x); %taille du fichier
B= 0.5;
delay=0.4;%retard(s)
m=delay*fs;
%algorithme écho (relation de récurrence)
y=zeros(size(x));%création d'une matrice pour les opération
for i=m+1:1:N;
    y(i)=x(i)+B*x(i-m);
end
sound(y,fs);%coute signal modifié
```

b)

```
%effet echo multiple
[x,fs]= audioread('ring03.wav');%lecture fichier.wav
sound(x,fs); %écoute du son original
N= length(x); %taille du fichier
A= 0.9;
B= 0.5;
delay=0.4;%retard(s)
m=delay*fs;
q=m;
%algorithme écho (relation de récurrence)
y=zeros(size(x));%création d'une matrice pour les opération
for i=m+1:1:N;
    y(i)=x(i)+A*x(i-q)+B*x(i-m);
end
sound(y,fs);%coute signal modifié
```

211.

a)

```
%effet reverberation simple
[x,fs]= audioread('ring03.wav');%lecture fichier.wav
sound(x,fs); %écoute du son original
N= length(x); %taille du fichier
B= 0.5;%atténuation
delay=0.4;%en seconde
m=delay*fs;
%algorithme écho (relation de récurrence)
y=zeros(size(x));%création d'une matrice pour les opération
for i=m+1:1:N;
    y(i)=x(i)+B*y(i-m);
end
sound(y,fs);%coute signal modifié
```

b)

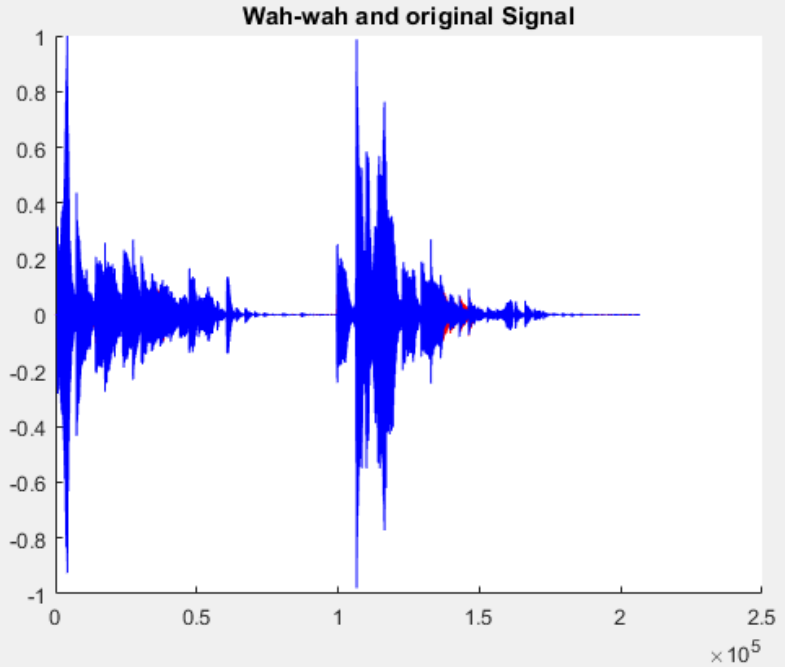
```
%reverberation version cathédrale
function[y]=fconv(x,h)
%x=fichier son, h=fichier impulsion
Ly=length(x)+length(h)-1;
Ly2= pow2(nextpow2(Ly));%recherché la plus êtite puissance de 2>Ly
X=fft(x,Ly2);%transformé en fourrier
H=fft(h,Ly2);%transformé de fourrier
Y=X.*H;%convolution
y=real(ifft(Y,Ly2)); %transformé de fourrier inverse
y=y(1:1:Ly);%N premier élements
y=y/max(abs(y));%sortie normalisée, retour de la fonction


%%
clear all
[x,Fs]=audioread('ring03.wav');
sound(x,Fs)
%lecture du fichier impulsion
[imp,Fsimp]=audioread('bonjour.wav');
%Do convolution with FFT
y=fconv(x,imp);
%write output
sound(y,Fs)
```

22.

221. WAH-WAH

```
[x,Fs]=audioread('Ring03.wav');%lecture du son orginal
sound(x,Fs)
%caractéristique du filtre passe-basse
damp=0.05;
minf=500;
maxf=3000;
%frequence du wah wah
Fw=2000;
%on fait varier la fréquence de manière triangulaire
delta=Fw/Fs;%pas
Fc=minf:delta:maxf;
while(length(Fc)<length(x))
    Fc=[Fc (maxf:-delta:minf)];
    Fc=[Fc (minf:delta:maxf)];
end
%adjustement de tableau des fréquence %
Fc=Fc(1:length(x));
%calcul des coefficients pour chaque Fc
F1=2*sin((pi*Fc(1))/Fs);
Q1=2*damp;
yh=zeros(size(x));%creation de vecteurs vides
yb=zeros(size(x));
yl=zeros(size(x));
%initialization du tableau pour éviter les valeurs negatives
yh(1)=x(1);
yb(1)=F1*yh(1);
yl(1)=F1*yb(1);
%calcul du signal filtré
for n=2:length(x),
```

```
    yh(n)=x(n)-yl(n-1)-Q1*yb(n-1);
    yb(n)=F1*yh(n)+yb(n-1);
    yl(n)=F1*yb(n)+yl(n-1);
    F1=2*sin((pi*Fc(n))/Fs);
end
%normalisation
maxyb=max(abs(yb));
yb=yb/maxyb;
figure(1)
hold on
plot(x,'r')
plot(yb,'b');sound(yb,Fs)
title('Wah-wah and original
Signal');
```



222. Vocodeur

a)Modulation d'amplitude par un signal sinusoïdal

```
[x,Fs]=audioread('Ring03.wav');
sound(x,Fs)
index=1:length(x);
Fc=440;
porteuse=sin(2*pi*index*(Fc/Fs))';
y=x.*porteuse;
sound(y,Fs)
```

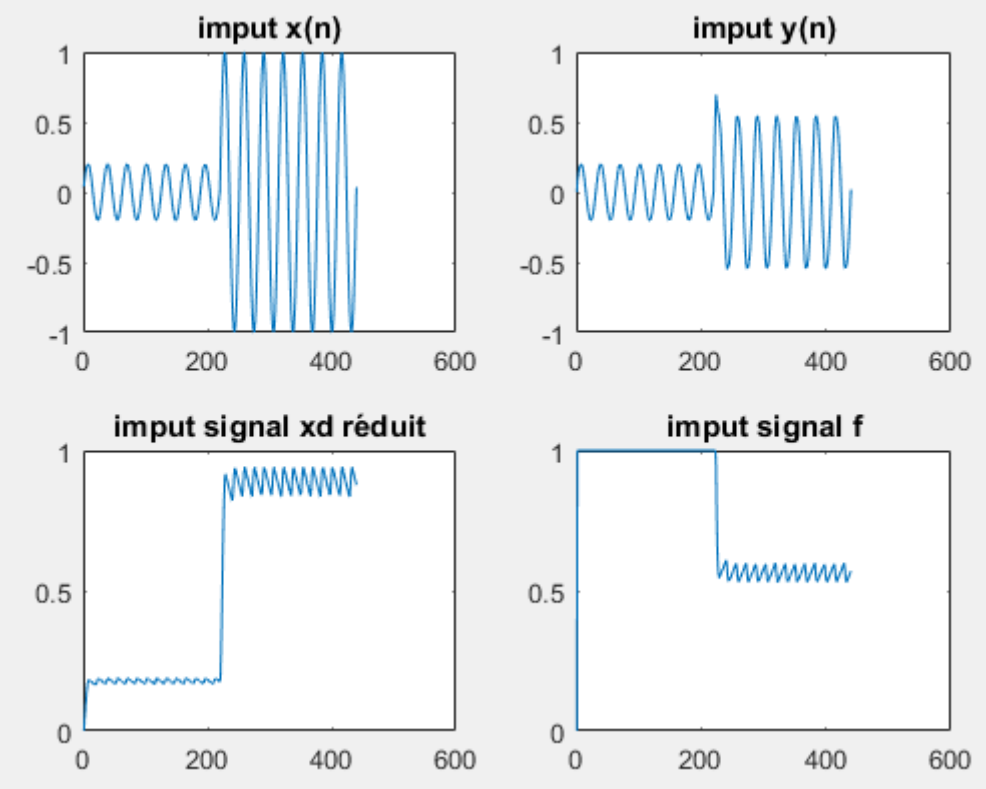
b) Modulation de phase et fréquence

```
[x,Fs]=audioread('Ring03.wav');
%sound(x,Fs)
index=1:length(x);
Fc=30;
y=cos((2*pi*index*(Fc/Fs))+x');
sound(y,Fs)%modulation de phase
z=diff(y)%modulation de fréquence
sound(z,Fs)
```

23. Effet du volume

232.

```
clear all
anzahl=220;
for n=1:anzahl,
    x(n)=0.2*sin(n/5)
end
for n=anzahl+1:2*anzahl;
    x(n)=sin(n/5);
end;
%seuillage
slope=1;
tresh=0.5;
rt=0.01;
at=0.4;
xd(1)=0;%enregistrement des pics dans x
for n=2:2*anzahl;
    a=abs(x(n))-xd(n-1);
    if a<0,a=0;end;
    xd(n)=xd(n-1)*(1-rt)+at*a
    if xd(n)>tresh,
        f(n)=10^(-slope*(log10(xd(n))-log10(tresh)));
        %calcul linéaire
else f(n)=1;
end;
y(n)=x(n)*f(n);
end;
subplot(2,2,1);plot(x);title('imput x(n)')
subplot(2,2,2);plot(y);title('imput y(n)')
subplot(2,2,3);plot(xd);title('imput signal xd réduit')
subplot(2,2,4);plot(f);title('imput signal f')
```



232Tremolo

A) Modulation d'amplitude à faible fréquence porteuse

```
[x,Fs]=audioread('Ring03.wav');
sound(x,Fs)
index=1:length(x);
Fc=5;
alpha=0.5;
trem=(1+alpha*sin(2*pi*index*(Fc/Fs)))';
y=trem.*x;
sound(y,Fs)
```

B) Modulation avec porteuse triangle

```
[x,Fs]=audioread('Ring03.wav');
%sound(x,Fs)
%createdu signal triangulaire
delta=5e-4;
minf=-0.5;
maxf=0.5;
trem=minf:delta:maxf;
while(length(trem)<length(x))
    trem=[trem (maxf:-delta:minf)];
    trem=[trem (minf:delta:maxf)];
end
trem=trem(1:length(x))';
%modulation avec le signaltriangulaire
y=x.*trem;
sound(y,Fs)
```

Music d'ambiance :

```
clear all;
[x,Fs]=audioread('Ring03.wav');
%lecture du fichier impulsion
[imp,Fsimp]=audioread('Ring03.wav');
%Do convolution with FFT
y=fconv(x,imp);
%write output
sound(y/4,Fs)
%%
[x,Fs]=audioread('Ring03.wav');
index=1:length(x);
Fc=440;
porteuse=sin(2*pi*index*(Fc/Fs))';
y=x.*porteuse;
sound(y,Fs)
%%
[x,Fs]=audioread('piano.wav');
index=1:length(x);
Fc=30;
y=cos((2*pi*index*(Fc/Fs))+x');
sound(y,Fs)%modulation de phase
z=diff(y)%modulation de fréquence
sound(z,Fs)
%%
```

```
[x,Fs]=audioread('Ring03.wav');
index=1:length(x);
Fc=30;
y=cos((2*pi*index*(Fc/Fs))+x');
sound(y,Fs)%modulation de phase
z=diff(y)%modulation de fréquence
sound(z,Fs)

%%
%effet echo simple
[x,fs]= audioread('ring03.wav');%lecture fichier.wav
%sound(x,fs); %écoute du son original
N= length(x); %taille du fichier
B= 0.5;
delay=0.4;%retard(s)
m=delay*fs;
%algorithme écho (relation de récurrence)
y=zeros(size(x));%création d'une matrice pour les opération
for i=m+1:1:N;
    y(i)=x(i)+B*x(i-m);
end
sound(y,fs);%coute signal modifié
```